

UNITED STATES PATENT APPLICATION

5

**METHOD AND APPARATUS FOR CREATING A MESSAGE DIGEST USING A
ONE-WAY HASH ALGORITHM**

10

INVENTOR:

Richard J. Takahashi

15

16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2

Method and Apparatus for Creating a Message Digest Using a One-Way Hash Algorithm

Technical Field of the Invention

5 The present invention relates generally to methods and apparatus for computing condensed representations of messages or data files, and more particularly to methods and apparatus for computing message digests using a one-way hash algorithm.

Background of the Invention

10 Hash functions have been widely used in modern cryptography to produce compressed data, message digests, fingerprints, and checksums, among other things. A hash function is a mathematical function that takes a variable-length input string, and
15 converts it to a fixed-length output string. The output string is called a hash value, which typically is smaller than the input string. A “one-way” hash function is a hash function that works in one direction, meaning that it is easy to compute a hash value from an input string, but it is difficult to generate a second input string that hashes to the same value. Bruce Schneier, Applied Cryptography, at 429-59 (1996) includes a detailed discussion
20 of various one-way hash algorithms.

A commonly used, one-way hash algorithm is “MD5”, where MD stands for “message digest.” MD5 was developed by Ron L. Rivest, and is described in detail in his paper entitled “The MD5 Message Digest Algorithm,” RFC 1321 (Apr. 1992).

When an arbitrarily large input message is input into MD5, the algorithm
25 produces a 128-bit output called a “fingerprint” or “message digest” of the input message. MD5 sequentially processes message blocks of 512 bits when computing a message digest. If a message is not a multiple of 512 bits, then MD5 first pads the message to make the message a multiple of 512 bits. The padded message is then processed by MD5 as n 512-bit blocks, M_1, \dots, M_n , where each block is composed of sixteen 32-bit words
30 or sub-blocks, $W_j, 0 \leq j \leq 15$. The main loop of MD5 processes each 512-bit block one at a time, and continues for as many 512-bit blocks as are in the message. The output of the algorithm is a set of four 32-bit words, which concatenate to form a single 128-bit

message digest. A four-word buffer (A, B, C, D) is used to compute the message digest, where each of A, B, C, and D is a 32-bit register, and the registers are initialized to particular values.

The main loop of MD5 has four “rounds,” r ($0 \leq r \leq 3$), where each round includes sixteen operations. Accordingly, sixty-four operations, i ($0 \leq i \leq 63$), are performed for each message block.

During each operation, a non-linear function (NLF) is performed on three of four 32-bit variables stored in A, B, C, and D. Then the operation adds the NLF output to the fourth variable, a sub-block, W_j , of the message, and a constant word, k_i . The operation then performs a left circular shift of a variable number of bits, s_i , and adds the result to the contents of one of A, B, C or D. Finally, that sum replaces the contents of one of A, B, C or D, and the next operation is performed. The NLF used for the operations in each round (i.e., each set of 16 sequential operations) is different from the NLF used in the previous round.

After the fourth round, the main loop repeats for the next message block, until the last block, M_n , has been processed. After processing the last block, the message digest is the 128-bit string represented by the concatenated words stored in A, B, C, and D.

MD5 can be performed by software or within an application specific integrated circuit (ASIC), where the operations are performed using hardware-implemented logic gates. Figure 1 illustrates a simplified, logical block diagram of one MD5 operation, in accordance with the prior art. Registers A, B, C, and D are represented by blocks 102, 104, 106, 108.

During one operation, a non-linear function 112 (NLF_r) is applied to three of the variables stored in registers A, B, C, and D. In the example round shown, the three variables input into NLF 112 are the variables stored in B 104, C 106, and D 108, although the input variables could differ for other rounds. The result is added, by a first full adder 114, to the contents of register A 102. A second full adder 116 adds the output of the first full adder 114 to the appropriate sub-block, W_j , for the round and operation being performed. A third full adder 118 then adds the output of the second full adder 116 to the appropriate constant word, k_i , for the round and operation being performed.

A shifter 120 then circularly left shifts the output of the third full adder 118 by the appropriate number of bits, s_i , for the round and operation being performed. Finally, the contents of register B 104 is added, by a fourth full adder 122, to the output of shifter 120. The output of full adder 122 is then added to the contents of register B 104, and that sum is placed in register A 102, for use during the next operation. The next operation will then use a different message sub-block, W_j , constant word, k_i , and number of shifts, s_i , in the left circular shift operation. In addition, the next operation will input the contents of different registers into NLF 112 and adders 114, 122. Finally, the result will be placed in a different register.

During the four rounds associated with one message block, the logic blocks illustrated in Figure 1 are cycled through sixty-four times. Further, the total number of cycles through the logic of Figure 1 is $64 \times n$, where n is the number of 512-bit blocks in the message. Each cycle through the logic corresponds to one clock cycle, and the clock frequency is limited by the various delays associated with the gates and other logical components. The logic depth of the operation illustrated in Figure 1 is rather substantial, because the logic includes computationally complex full adders, among other elements. Accordingly, the cumulative delay associated with this design is long, and consequently the clock frequency must be fairly low.

As the desire to compress data faster increases, communication systems increasingly place more stringent demands on the computation speed of cryptographic algorithms. Accordingly, what are needed are one-way hash algorithms and apparatus, which produce the same output as MD5 in less time. Further, what are needed are an MD5 compatible hash algorithm and apparatus, which have less logic depth than the standard MD5 implementation.

Brief Description of the Drawing

Figure 1 illustrates a simplified, logical block diagram of one MD5 operation, in accordance with the prior art;

Figure 2 illustrates a simplified, logical block diagram corresponding to one round of sixteen operations, in accordance with one embodiment of the present invention;

Figure 3 illustrates a flowchart of a method for creating a message digest, in accordance with one embodiment of the present invention; and

Figure 4 illustrates an electronic device in which the embodiments of the invention may be practiced, in accordance with one embodiment of the present invention.

5

Detailed Description of the Invention

Various embodiments of the present invention provide a one-way hash algorithm and apparatus, which produce the identical message digest as MD5, given the same input message, but in nearly half the time necessary using the standard MD5 implementation. In various embodiments, this is accomplished by separately computing part of the first operation of a 16-operation round using a front computation process, and computing the remaining operations of the round using a very fast systolic computation process. Because it has fewer delays in its logic, the systolic computation process can be performed using a clock rate that is approximately twice the clock rate possible using the standard MD5 implementation. In addition, the systolic computation process computes portions of adjacent operations in parallel. Accordingly, the time to compute each 16-operation round is reduced, using the various embodiments, to nearly half the time necessary to compute each 16-operation round using the standard MD5 implementation.

Similar to MD5, when an input message of any arbitrary length of bits is input into the algorithm of one of the various embodiments, the algorithm produces a 128-bit output, referred to herein as a message digest. Although the term "message digest" has been used to indicate the output result of the algorithm, such terminology is not meant to limit the various embodiments to specific applications.

In one embodiment, the method of the present invention sequentially processes blocks of 512 bits when computing a message digest. If a message is not a multiple of 512 bits, then the algorithm first pads the message to make the message a multiple of 512 bits.

The padded message is then processed by MD5 as n 512-bit blocks, M_1, \dots, M_n , where each block is composed of sixteen 32-bit words or sub-blocks, W_j ($0 \leq j \leq 15$). The main loop of MD5 processes each 512-bit block one at a time, and continues for as

many 512-bit blocks as are in the message. The output of the algorithm is a set of four 32-bit words, which concatenate to form a single 128-bit message digest.

A four-word buffer (A, B, C, D) is used to compute the message digest, where each of A, B, C, and D is a 32-bit register. These registers are initialized to particular values, which are the same initialization values as are used in the standard MD5 implementation.

As described previously, the main loop of MD5 has four rounds, r ($0 \leq r \leq 3$), where each round includes sixteen operations. Accordingly, sixty-four operations, i ($0 \leq i \leq 63$), are performed for each message block. An "operation" is defined herein as a set of processes that operates on a word of the sequence of input words derived from the message. In one embodiment, the set of processes associated with each operation are described in the next paragraph. For one word, the set of processes culminates in the replacement of one of the registers A, B, C or D with a result that is calculated during the set of processes for that word.

During each operation, the set of processes includes the following, in one embodiment. First, a non-linear function (NLF) is performed on three of four 32-bit variables stored in A, B, C, and D. Then the operation adds the NLF output to the fourth variable, a sub-block, W_j , of the message (i.e., a word of the message), and a constant word, k_i . The operation then performs a left circular shift of a variable number of bits, s_i , and adds the result to the contents of one of A, B, C or D. Finally, that sum replaces the contents of one of A, B, C or D.

In one embodiment, part of the first operation of a 16-operation round is separately computed using a front computation process, and the remaining operations of the round are computed using a very fast systolic computation process. This systolic computation process, which is described in detail below, performs portions of consecutive operations in parallel. This parallel processing, along with a more shallow logic depth per operation, enable the time to compute each round to be substantially reduced, because a faster clock frequency can be used for the systolic computation process.

The NLF used for the operations in each round (i.e., each set of 16 sequential operations) is different from the NLF used in the previous round. Each NLF takes as

input three 32-bit words and produce as output one 32-bit word. The four NLFs are defined as follows, and are the same as the NLFs used in the standard MD5 implementation:

- 5 $F(X,Y,Z) = (X \text{ AND } Y) \text{ OR } ((\text{NOT } X) \text{ AND } Z)$ (for round 1: $0 \leq i \leq 15$)
 $G(X,Y,Z) = (X \text{ AND } Z) \text{ OR } (Y \text{ AND } (\text{NOT } Z))$ (for round 2: $16 \leq i \leq 31$)
 $H(X,Y,Z) = X \text{ XOR } Y \text{ XOR } Z$ (for round 3: $32 \leq i \leq 47$)
 $I(X,Y,Z) = Y \text{ XOR } (X \text{ OR } (\text{NOT } Z))$ (for round 4: $48 \leq i \leq 63$).

- 10 The main loop of the algorithm is performed as described below. First, the values in the four registers of the buffer (A, B, C, D) are retained and copied into four 32-bit variables a, b, c, and d, so that $a = A$, $b = B$, $c = C$, and $d = D$.

- Each of the four rounds is then performed by applying the following logic, which is the same logic as is used in the standard MD5 implementation. In the functions below,
15 W_j represents the j th sub-block of the message ($0 \leq j \leq 15$), $\lll s_i$ represents a left circular shift of s bits, and "+" denotes the addition of words. The actual values for W_j , k_i , and s_i for each operation, i , are found in the "List of Operations" table, below.

 Round 1: For $i = 0$ to 15,

- FF(a,b,c,d, W_j , s_i , k_i) denotes the operation
20 $a = b + ((a + F(b,c,d) + W_j + k_i) \lll s_i)$.

 Round 2: For $i = 16$ to 31,

 GG(a,b,c,d, W_j , s_i , k_i) denotes the operation
 $a = b + ((a + G(b,c,d) + W_j + k_i) \lll s_i)$.

 Round 3: For $i = 32$ to 47,

- 25 HH(a,b,c,d, W_j , s_i , k_i) denotes the operation
 $a = b + ((a + H(b,c,d) + W_j + k_i) \lll s_i)$.

 Round 4: For $i = 48$ to 63,

 II(a,b,c,d, W_j , s_i , k_i) denotes the operation
 $a = b + ((a + I(b,c,d) + W_j + k_i) \lll s_i)$.

30

During each round, r ($0 \leq r \leq 3$), the three variables operated upon by the NLF, the message sub-block, W_j , the constant word, k_i , and the number of shifts, s_i , in the left circular shift operation change from operation to operation. For each round and operation, these operations are performed sequentially as follows, where the operations listed below are identical to the 64 operations used in the standard MD5 implementation. The operation number, i ($0 \leq i \leq 63$), are listed to the left of each operation.

List of Operations

Round 1 ($r = 0$):

- | | |
|----|--|
| 10 | 0. FF(a,b,c,d, W_0 , 7, d76aa478) |
| | 1. FF(d,a,b,c, W_1 , 12, e8c7b756) |
| | 2. FF(c,d,a,b, W_2 , 17, 242070db) |
| | 3. FF(b,c,d,a, W_3 , 22, c1bdceee) |
| | 4. FF(a,b,c,d, W_4 , 7, f57c0faf) |
| 15 | 5. FF(d,a,b,c, W_5 , 12, 4787c62a) |
| | 6. FF(c,d,a,b, W_6 , 17, a8304613) |
| | 7. FF(b,c,d,a, W_7 , 22, fd469501) |
| | 8. FF(a,b,c,d, W_8 , 7, 698098d8) |
| | 9. FF(d,a,b,c, W_9 , 12, 8b44f7af) |
| 20 | 10. FF(c,d,a,b, W_{10} , 17, ffff5bb1) |
| | 11. FF(b,c,d,a, W_{11} , 22, 895cd7be) |
| | 12. FF(a,b,c,d, W_{12} , 7, 6b901122) |
| | 13. FF(d,a,b,c, W_{13} , 12, fd987193) |
| | 14. FF(c,d,a,b, W_{14} , 17, a679438e) |
| 25 | 15. FF(b,c,d,a, W_{15} , 22, 49b40821) |

Round 2 ($r = 1$):

- | | |
|----|--|
| | 16. GG(a,b,c,d, W_1 , 5, f61e2562) |
| | 17. GG(d,a,b,c, W_6 , 9, c040b340) |
| 30 | 18. GG(c,d,a,b, W_{11} , 14, 265e5a51) |
| | 19. GG(b,c,d,a, W_0 , 20, e9b6c7aa) |

20. GG(a,b,c,d, W₅, 5, d62f105d)
21. GG(d,a,b,c, W₁₀, 9, 02441453)
22. GG(c,d,a,b, W₁₅, 14, d8a1e681)
23. GG(b,c,d,a, W₄, 20, e7d3fbc8)
- 5 24. GG(a,b,c,d, W₉, 5, 21e1cde6)
25. GG(d,a,b,c, W₁₄, 9, c33707d6)
26. GG(c,d,a,b, W₃, 14, f4d50d87)
27. GG(b,c,d,a, W₈, 20, 455a14ed)
28. GG(a,b,c,d, W₁₃, 5, a9e3e905)
- 10 29. GG(d,a,b,c, W₂, 9, fcefa3f8)
30. GG(c,d,a,b, W₇, 14, 676f02d9)
31. GG(b,c,d,a, W₁₂, 20, 8d2a4c8a)

Round 3 (r = 2):

- 15 32. HH(a,b,c,d, W₅, 4, fffa3942)
33. HH(d,a,b,c, W₈, 11, 8771f681)
34. HH(c,d,a,b, W₁₁, 16, 6d9d6122)
35. HH(b,c,d,a, W₁₄, 23, fde5380c)
36. HH(a,b,c,d, W₁, 4, a4beea44)
- 20 37. HH(d,a,b,c, W₄, 11, 4bdecfa9)
38. HH(c,d,a,b, W₇, 16, f6bb4b60)
39. HH(b,c,d,a, W₁₀, 23, bebfbc70)
40. HH(a,b,c,d, W₁₃, 4, 289b7ec6)
41. HH(d,a,b,c, W₀, 11, eaa127fa)
- 25 42. HH(c,d,a,b, W₃, 16, d4ef3085)
43. HH(b,c,d,a, W₆, 23, 04881d05)
44. HH(a,b,c,d, W₉, 4, d9d4d039)
45. HH(d,a,b,c, W₁₂, 11, e6db99e5)
46. HH(c,d,a,b, W₁₅, 16, 1fa27cf8)
- 30 47. HH(b,c,d,a, W₂, 23, c4ac5665)

Round 4 ($r = 3$):

- 48. $\Pi(a,b,c,d, W_0, 6, f4292244)$
- 49. $\Pi(d,a,b,c, W_7, 10, 432aff97)$
- 50. $\Pi(c,d,a,b, W_{14}, 15, ab9423a7)$
- 5 51. $\Pi(b,c,d,a, W_5, 21, fc93a039)$
- 52. $\Pi(a,b,c,d, W_{12}, 6, 655b59c3)$
- 53. $\Pi(d,a,b,c, W_3, 10, 8f0ccc92)$
- 54. $\Pi(c,d,a,b, W_{10}, 15, ffeff47d)$
- 55. $\Pi(b,c,d,a, W_1, 21, 85845dd1)$
- 10 56. $\Pi(a,b,c,d, W_8, 6, 6fa87e4f)$
- 57. $\Pi(d,a,b,c, W_{15}, 10, fe2ce6e0)$
- 58. $\Pi(c,d,a,b, W_6, 15, a3014314)$
- 59. $\Pi(b,c,d,a, W_{13}, 21, 4e0811a1)$
- 60. $\Pi(a,b,c,d, W_4, 6, f7537e82)$
- 15 61. $\Pi(d,a,b,c, W_{11}, 10, bd3af235)$
- 62. $\Pi(c,d,a,b, W_2, 15, 2ad7d2bb)$
- 63. $\Pi(b,c,d,a, W_9, 21, eb86d391)$

After round 4 has been completed, a, b, c, and d are added to the retained contents of A, B, C, and D (i.e., the contents before this message block was processed), respectively. The main loop then repeats for the next message block, until the last block, M_n , has been processed. After processing the last block, the message digest is the 128-bit string represented by the concatenated words stored in A, B, C, and D.

Figure 2 illustrates a simplified, logical block diagram corresponding to one round of sixteen operations, in accordance with one embodiment of the present invention. The logic for performing one round includes two major logic blocks: a front computation block 202 and a systolic computation block 204. The variable t ($0 \leq t \leq 15$) indicates which operation is being performed within a round, r ($0 \leq r \leq 3$). Accordingly, the operation number, i , is equal to $t \times r$.

The front computation block 202 performs a portion of the first operation (i.e., $t = 0$) for each round. The systolic computation block 204 performs a remainder of the first

operation, and each of the fifteen subsequent operations (i.e., $1 \leq t \leq 15$) that comprise the round. In one embodiment, the front computation block 202 is clocked at a first clock frequency (e.g., 200MHz), and the systolic computation block 204 is clocked at a second, faster clock frequency (e.g., 400MHz).

5 The first clock frequency can be approximately equal to the clock frequency used in a standard MD5 implementation. This first clock frequency is limited, on the high end, by the delays associated with the logic within the front computation block 202. The second clock frequency can be approximately equal to twice the clock frequency used in a standard MD5 implementation, because the delays associated with the systolic
10 computation block 204 are substantially less than the delays associated with performing one standard MD5 operation. Accordingly, the various embodiments of the present invention are able to compute one 16-operation round in nearly half the time it takes to compute a round using a standard MD5 implementation. Where a standard MD5 implementation uses 64 clock cycles at a particular clock frequency, the various
15 embodiments of the present invention use the equivalent of approximately 34 clock cycles at the particular clock frequency.

 In one embodiment, it takes one clock cycle to complete the front computation portion 202, and one clock cycle (potentially at a higher frequency) to complete each iteration through the systolic computation portion 204. In other embodiments, it could
20 take more than one clock cycle to complete either or both the front computation and/or systolic computation portions. In other words, during a particular round, the front computation portion 202 is performed during one or more clock cycles, and each iteration of the systolic computation portion 204 is performed during one or more subsequent clock cycles.

25 Referring again to Figure 2, registers A, B, C, and D are represented by blocks 206, 208, 210, 212. When a round begins, the front computation portion 202 is performed first.

 During the front computation portion 202, a first non-linear function block 214 (NLF_r) is applied to the contents of registers B 208, C 210, and D 212. During the first
30 round ($r = 0$), the appropriate NLF to use is $F(X, Y, Z) = (X \text{ AND } Y) \text{ OR } ((\text{NOT } X) \text{ AND } Z)$, where $X = B$, $Y = C$, and $Z = D$. A first carry save adder 216 (CSA) adds the output

of NLF 214 to W_j and k_i . For example, during the first operation of the first round, $W_j = W_0$ and $k_i = d76aa478$ (See the "List of Operations" table, above). First CSA 216 is a three-input/two-output carry save adder, in one embodiment. In other embodiments, multiple CSAs and/or full adders could be used to add the three inputs.

5 A second CSA 218 then adds the output of the first CSA 216 to the contents of register A 206. In one embodiment, second CSA 218 also is a three-input/two-output carry save adder, although multiple CSAs and/or full adders could be used to add the three inputs, in other embodiments.

10 During the front computation portion 202, a variable, v , is set to a value of 1. The variable roughly indicates which operation, within a particular round, is being performed. Although the first operation has not yet been completed, setting v to the value of 1 enables the logic in the systolic computation portion 204 to perform correctly.

15 The systolic computation portion 204 is then performed. First, a multiplexer 220, is used to select, as input, either the output of the front computation portion 202, or values generated within the systolic computation portion 204. When $v = 1$, multiplexer 220 selects the output of the front computation portion 202, which includes a sum and a carry produced by CSA 218. Multiplexer 220 then passes the sum and carry to a first full adder 222. When $v > 1$, multiplexer 220 selects a stored output of a second full adder 232 and a stored result produced by a second NLF block 236, and passes those values to
20 the first full adder 222.

25 The first full adder 222 adds the two values received from multiplexer 220. Shifter 224 then circularly left shifts the output of first full adder 222 by a variable number of bits, s_i , which has a value that depends on the number of the operation, i , being performed. For example, during the first operation, $i = 0$, of the first round, $r = 0$, $s_i = 7$ bits. During the second operation, $i = 1$, of the first round, $s_i = 12$ bits, and so on (See the "List of Operations" table, above). In one embodiment, shifter 224 is implemented as a hard-wired shift. In another embodiment, shifter 224 is implemented through logic, or is performed using software. Although a left circular shift is used in one embodiment, the same result could also be achieved using a right circular shift of a complementary number
30 of bits.

A third full adder 226 then adds the output of shifter 224 to the contents of register B 208. The output of third full adder 226 represents the final result of an operation. Accordingly, for example, if the operation is the first operation of a first round, the output represents $FF(a,b,c,d,W_0,7,d76aa478)$. If the operation is the second operation of the first round, the output represents $FF(d,a,b,c,W_1,12,e8c7b756)$ (See the “List of Operations” table, above).

The output of third full adder 226 corresponds to the result produced by full adder 122 (Figure 1) used in a standard MD5 implementation. As Figure 1 indicates, in the standard MD5 implementation, that result is added to the contents of register A 102 (Figure 1). However, in one embodiment of the present invention, the result is temporarily stored as variable BNEW 228. The later incorporation of BNEW 228 into the four-word register A, B, C, and D will be explained below.

In one embodiment, a portion of the next operation is computed in parallel with blocks 220-226, which represent a portion of the preceding operation. Specifically, a third CSA 230 is used to add the contents of register D 212 to W_j and k_{i+1} . For example, during the second operation of the first round, $W_j = W_1$ and $k_{i+1} = e8c7b756$ (See the “List of Operations” table, above). Third CSA 230 is a three-input/two-output carry save adder, in one embodiment. In other embodiments, multiple CSAs and/or full adders could be used to add the three inputs. A second full adder 232 adds the outputs of the third CSA 230 (i.e., it incorporates the carry into the sum). The output of the second full adder 232 is temporarily held in register TEMP1 234, until the systolic computation block 204 is clocked for the next cycle.

Another portion of the next operation is performed when a second NLF block 236 is applied to the output of the third full adder 226 (i.e., BNEW), and the contents of register B 208 and register C 210. The appropriate NLF to use is the same as the NLF used for the first NLF block 214. Accordingly, during the first round, the appropriate NLF to use is $F(X,Y,Z) = (X \text{ AND } Y) \text{ OR } ((\text{NOT } X) \text{ AND } Z)$, where $X = \text{BNEW}$, $Y = B$, and $Z = C$. The output of NLF 236 is temporarily held in register TEMP2 238, until the systolic computation block 204 is clocked for the next cycle.

Concurrently with clocking the systolic computation block 204 for the next cycle, the contents of registers B 208, C 210, D 212, and A 206 are replaced, so that $A = D$, $D =$

C, C = B, and B = BNEW 128. These new register values are then available for the next cycle.

The variable v is then incremented by 1. If v is less than 16, but greater than 1, then multiplexer 220 selects, as input, the values within TEMP1 234 and TEMP2 238, and the systolic computation block 204 completes the next operation, as described above. The procedure then iterates until all of the round's operations have been performed. If v = 16, then the front computation portion 202 is again performed for the next round, as described above, and the procedure iterates until all four rounds have been performed. After adding the results from the four rounds to retained contents of A, B, C, and D, the entire procedure then iterates for each remaining message block. Once all message blocks have been processed, the message digest is the concatenation of the contents of registers A, B, C, and D. In one application, the message digest can then be input into a verification or signature algorithm (e.g., DSA), or can otherwise be stored, transmitted, or used to compute a value that has some usefulness.

The above description indicates that a portion of one operation is performed in parallel with a portion of a next operation (i.e., some of the processes involved in each operation are performed in parallel). The remaining processes of the next operation are performed later. When the embodiment is implemented using logic, the portion of one operation and the portion of the next operation are completed during a first clock cycle, and the remaining processes of the next operation are completed during the next clock cycle. When one of the operations is the first operation of a round, the initial portion of the operation is performed during a preceding clock cycle (i.e., it is performed before performing the remaining processes of the first operation).

Using the embodiment shown in Figure 2, the critical path through the majority of operations ($1 \leq t \leq 15$) includes multiplexer 220, shifter 224, two full adders 222 and 226, NLF 236, and register setup time. Referring to Figure 1, using a conventional MD5 implementation, the critical path for each operation includes NLF 112, four full adders 114, 116, 118, 122, shifter 120, and register setup time. Because the critical path for the embodiment shown in Figure 2 includes only two full adders for most operations, as opposed to four full adders in the critical path for a conventional MD5 implementation, the logic depth and the amount of time to process a full message is substantially reduced

from the conventional MD5 implementation. With nearly half the logic depth, and the ability to roughly double the clock frequency during the systolic portion 204 of each round, the embodiments of the present invention can compute a message digest in approximately one quarter the time necessary for a conventional MD5 implementation to compute a message digest.

Figure 3 illustrates a flowchart of a method for creating a message digest, in accordance with one embodiment of the present invention. It would be obvious to one of skill in the art, that the method could be entirely or partially accomplished in an integrated circuit (e.g., an ASIC) and/or by software.

The method begins, in block 302, by padding the message for which a message digest is to be computed, if necessary. As described previously, if a message is not a multiple of 512 bits, then the method first pads the message with a single "1" and as many zeros are necessary to make the message a multiple of 512 bits, except that the last 64 bits of the last 512-bit block are reserved for the length, l , of the original message.

The padded message is then processed by the algorithm as n 512-bit blocks, M_1, \dots, M_n .

In block 304, registers A, B, C, and D are initialized. In one embodiment, these registers are initialized to be the same values as the predetermined set of initialization values used in MD5. These values are as follows, in hexadecimal:

A = 01234567

B = 89abcdef

C = fedcba98

D = 76543210

The outside loop of the algorithm (blocks 306-338), which sequentially selects each message block, M_1, \dots, M_n , then begins. In block 306, the next message block, M_x , is selected for processing, and divided into sixteen 32-bit words, W_0, W_1, \dots, W_{15} , where W_0 is the left-most word. During the first iteration of the outside loop that includes blocks 306-338, the "next block" is block M_1 . In block 308, a variable r , which indicates which round the algorithm is computing, is then set to a value of 0, and an operation variable i , which indicates which operation is being performed of the 64 operations per

message block, is also set to a value of 0. In block 309, the contents of registers A, B, C, and D are retained for later use.

The middle loop of the algorithm (blocks 310-336), which steps through the four rounds for one message block, then begins. Referring also to Figure 2, the front computation process is initiated, which begins the calculations for the first operation within a round. First, in block 310, a first NLF (e.g., NLF_r 214) is applied to the contents of registers B 208, C 210, and D 212. Next, in block 312, the sum of the NLF output, W_j , k_i , and the contents of register A 206 is computed. For ease of description, this sum is referred to as "SUM0." In one embodiment, SUM0 is computed using one or more carry save adders (e.g., adders 216, 218, Figure 2). The front computation process is completed, in one embodiment, by setting the variable v to a value of 1, in block 314, where the variable v roughly indicates which operation, within a particular round, is being performed.

The inner loop of the algorithm (blocks 316-332), which represents the systolic computation process (e.g., block 204, Figure 2) of a particular round, then begins. First, the variable v is evaluated, in block 316, to determine if it is equal to 1. In one embodiment, this evaluation is performed by multiplexer 220 (Figure 2). If v is equal to 1, then the multiplexer selects the outputs of the front computation process (e.g., the outputs of adder 218, Figure 2) as the inputs to the inner loop calculations. If v is not equal to 1, then the multiplexer selects the outputs of other inner loop components (e.g., registers TEMP1 234 and TEMP2 238, Figure 2) as the inputs to the inner loop calculations.

The sum of the multiplexer outputs is calculated in either block 318 or 320. For ease of description, this sum is referred to as "SUM1." If v equals 1, then SUM1 = SUM0 is calculated, in block 318. Thus, in one embodiment, SUM1 represents the sum calculated by the front computation process after the carry has been incorporated. If v is not equal to 1, then SUM1 = $NLF_r(BNEW) + SUM2$ is calculated, in block 320. As will be described below, SUM2 = $W_j + k_{i+1} + D$ was pre-calculated during a previous round (see block 325). In one embodiment, SUM1 represents the sum of TEMP1 234 and TEMP2 238 (Figure 2), and SUM1 is calculated by a full adder (e.g., adder 222, Figure 2).

In block 322, SUM1 is then circularly left shifted by the appropriate number of bits, s_i , for the operation. For ease of description the shifted version of SUM1 is referred to as "RESULT." In one embodiment, the circular shift is performed by a shifter (e.g., shifter 224, Figure 2).

5 RESULT is then added, in block 324, to the contents of register B (208, Figure 2). In one embodiment, this sum is calculated by a full adder (e.g., adder 226, Figure 2), and is temporarily stored in a register (e.g., register 228, Figure 2) as variable BNEW. BNEW is the result of the end of an operation, and will later be stored in register 208 (Figure 2), after the registers have been rotated.

10 A portion of a next operation is calculated in parallel with any or all of blocks 316-326, by calculating "SUM2," which equals the sum of W_j , k_{i+1} , and the contents of register D 212 (Figure 2), in block 325. In one embodiment, this calculation is performed by adders 230, 232 (Figure 2). By pre-calculating this value for the next operation in parallel with the current operation, the critical path for each operation performed in the
15 systolic computation portion is reduced. Accordingly, the time to calculate each round is reduced. Another portion of the next operation is then calculated, in block 326, when a second NLF (e.g., NLF_r 236, Figure 2) is applied to BNEW, and the contents of registers B 208 and C 210 (Figure 2).

In one embodiment, the operation variable, i , is then incremented by 1, in block
20 327. In other embodiments, i could be incremented at an earlier or later time. The registers A, B, C, and D are then rotated or replaced, in block 328, so that $A = D$, $D = C$, and $C = B$, and $B = BNEW$ (i.e., the result from the previous operation). The variable v is then incremented by 1, in block 330, and a determination is made, in block 332, whether the variable $v = 16$. If not, it indicates that the round has not yet completed, and
25 the procedure repeats the inner loop, as shown. Specifically, the multiplexer will be called upon to pass the values through that are necessary to perform the next operation. If the variable $v = 16$, it indicates that the round is completed.

If the round is completed, then, in block 337, the contents of registers A, B, C, and D are added to the values previously retained in block 309. A determination is then
30 made, in block 334, whether the variable r is less than 3. If so, it indicates that one or more rounds must still be completed for the message block. In that case, the variable r is

incremented by 1 in block 336, and the procedure repeats the middle loop, as shown. Specifically, the front computation process will again be performed for the next round's first operation, and then the systolic computation process will be performed for the remaining fifteen operations.

5 If the fourth round is completed, a determination is made, in block 338, whether all message blocks, M_1, \dots, M_n , that comprise the message have been processed. If not, then the process repeats the outer loop, as shown. Specifically, the next message block is selected and divided, and the four rounds (i.e., 64 operations) are performed for the next message block. If all message blocks have been processed, then the method ends.

10 The above description indicates that the algorithm operates on input words, specifically 32-bit words. In other embodiments, the algorithm could be adapted to operate on larger or smaller words. In addition, in one embodiment, the algorithm and/or the system within which the algorithm operates could be adapted to receive message bits in a serial manner, rather than a parallel manner. In such an embodiment, a sequence of
15 serial bits could be fed into one or more registers (e.g., registers A, B, C, and D, or other registers), and once the register is filled to the register size, the word could be processed as described above. The next set of serial bits would then be loaded into the register, and the process would repeat. Accordingly, in one embodiment, the algorithm could include performing a serial to parallel conversion process, prior to performing a round that
20 operates on the set of serial bits that comprise a word.

 In one embodiment, some or all of the algorithm operations are performed within an ASIC, where the operations are performed using logic. In other embodiments, some or all of the algorithm operations are performed using software.

 The various embodiments could be used in many different types of devices. For
25 example, they could be used in wired or wireless communication devices (e.g., radios, pagers, cellular or conventional telephones), "smart cards," PCICM cards, access tokens, routers, switches, and any other device that utilizes a one-way hash algorithm. These examples are provided for purposes of illustration and are not intended to limit the use of the various embodiments in other applications.

30 The message to be processed could originate at a particular device. For example, the message could be stored within a device or could be generated in real time by the

device (e.g., voice data from the device's user). Alternatively, the message could be received from a remote device. In addition, the message digest calculated using the various embodiments could be stored, used or consumed internally by a device, or it could be transmitted to another device for storage and/or processing.

5 Figure 4 illustrates an electronic device in which the embodiments of the invention may be practiced, in accordance with one embodiment of the present invention. Device 400 includes integrated circuit 402, computer readable storage medium 404, and external interface 406, in one embodiment.

10 When all or part of the methods of the various embodiments are implemented in hardware, integrated circuit 402 includes one or more ASICs, each of which include the logic for performing all or part of the one-way hash function (e.g., the front computation logic block 202 and the systolic computation logic block 204, Figure 2). In such an embodiment, device 400 may also include a processor (not shown), which places the input message block in a format that is useable by the ASIC. For example, a processor
15 may be used to pad the message, divide the message into blocks, and/or initialize various registers. The A, B, C, D registers could be implemented in integrated circuit 402, a processor, computer readable storage medium 404, or another device.

20 The message and/or message blocks could be stored in a memory device, such as computer readable storage medium 404, or the message and/or message blocks could be received through external interface 406. Computer readable storage medium 404 could be, for example, RAM, ROM, hard drive, CD, magnetic disk, disk drive, a combination of these types of storage media, and/or other types of storage media that are well known to those of skill in the art. When all or part of the methods of the various embodiments are implemented in software, computer readable storage medium 404 also could be used
25 to store computer executable instructions, which carry out all or part of the methods, when executed. In such an embodiment, integrated circuit 402 could be a microprocessor, ASIC or another type of integrated circuit capable of executing the computer executable instructions. In other embodiments, where storage of computer executable instructions, message data, message digests, or other data is not necessary,
30 device 400 may not include storage medium 404.

External interface 406 could be, for example, a user interface (e.g., a keyboard, speaker, or other input device) or an interface to a wired or wireless external network, system or device. External interface 406 could be used to receive input messages and/or message blocks, and/or could be used to transmit or receive message digests, digital
5 signatures, or verification or other data that was generated using an embodiment of the present invention. Data received and/or transmitted by external interface 406 could be sent to or received from, respectively, either or both integrated circuit 402 and/or storage medium 404. In other embodiments, where transmission or receipt of message data, message digests or other data is not necessary, device 400 may not include external
10 interface 406.

Conclusion

Various embodiments of a one-way hash algorithm have been described. The
15 various embodiments can be used to produce a message digest that is identical to a message digest produced by MD5, given the same input message. However, the algorithms of the various embodiments produce the message digest using the equivalent of approximately half the clock cycles and less logic depth than MD5.

In the foregoing detailed description, reference is made to the accompanying
20 drawings, which form a part hereof, and in which are shown by way of illustration specific embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention.

It will be appreciated by those of ordinary skill in the art that any arrangement, which is calculated to achieve the same purpose, may be substituted for the specific
25 embodiment shown. In addition, although certain applications of the embodiments have been listed above, the embodiments could be incorporated into any other application that could benefit from the use of a one-way hash algorithm. The various embodiments could also be used, with or without modifications, as compatible, alternative implementations of other hash algorithms. For example, but not by way of limitation, the embodiments
30 could be used as compatible algorithms to future MD5 implementations. Therefore, all

such applications and alternative implementations are intended to fall within the spirit and scope of the present invention.

This application is intended to cover any adaptations or variations of the present invention. The foregoing detailed description is, therefore, not to be taken in a limiting
5 sense, and it will be readily understood by those skilled in the art that various other changes in the details, materials, and arrangements of the parts and steps, which have been described and illustrated in order to explain the nature of this invention, may be made without departing from the spirit and scope of the invention as expressed in the adjoining claims.

FOR OFFICIAL USE ONLY